UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/822,390	04/12/2004	Lawrence A. Spracklen	004-40364	2873

42714 7590 12/05/2006

SUN MICROSYSTEMS, INC.
ATTN: TIMOTHY SCHULTE
ONE STORAGETEK DRIVE, MS 4309
LOUISVILLE, CO 80028-4309

EXAMINER	
MOLL, JESSE R	

ART UNIT	PAPER NUMBER
2181	

DATE MAILED: 12/05/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/822,390	SPRACKLEN ET AL.	
	Examiner	Art Unit	
	Jesse R. Moll	2181	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 September 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-66* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *58-61* is/are allowed.

6)☒ Claim(s) *1-3,10-57 and 62-64* is/are rejected.

7)☒ Claim(s) *4-9,65 and 66* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08 September 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

FRITZ FLEMING
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Withdrawn Objections / Rejections

1. Applicant, via amendment has overcome the rejection of claims 48-53 under 35 USC 101. The rejection has been respectfully withdrawn.

2. Applicant, via amendment has overcome the objection to the specification. The objection has been respectfully withdrawn.

3. Applicant, via amendment has overcome the rejections under 35 USC 112 second paragraph. The rejections have been respectfully withdrawn.

Drawings

4. The drawings were received on 8 September 2006. These drawings are acceptable.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 16-47 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims to computer-related inventions that are clearly nonstatutory fall into the same general categories as nonstatutory claims in other arts, namely natural phenomena such as magnetism, and abstract ideas or laws of nature which constitute "descriptive material. "Abstract ideas, Warmerdam, 33 F.3d at 1360, 31 USPQ2d at 1759, or the mere manipulation of abstract ideas, Schrader, 22 F.3d at 292-93, 30 USPQ2d at 1457-58, are not patentable. Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." In this context, "functional descriptive material" consists of data structures and computer programs which impart functionality when employed as a computer component. (The definition of "data structure" is "a physical or logical relationship among data elements, designed to support specific data manipulation functions." The New IEEE Standard Dictionary of Electrical and Electronics Terms 308 (5th ed. 1993).) "Nonfunctional descriptive material" includes but is not limited to music, literary works and a compilation or mere arrangement of data (See MPEP section 2106, IV, B, 1).

Claim 16 comprises steps of tracking and predicting. The steps are just an abstract idea. The claims do not provide practical application that produces a useful, tangible and concrete result. Therefore, the claim is non-statutory. Similar problems exist in the other claims 28, and 38.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

8. Claims 1-3, 10-47, 54-57, and 62-64 are rejected under 35 U.S.C. 102(a) as being anticipated by Shen et al., "*Modern Processor Design-Fundamentals of Superscalar Processors*", 2003, McGraw-Hill Higher Education, Beta edition, Chapter 4, pages 196-202, herein referred to as Shen et al.

Referring to claim 1, Shen et al. discloses, as claimed, a processor (Fig. 4-46) that predicts aliasing (see Page 201, lines 9-14, regarding alias prediction) between read type instructions (load instructions) and write type instructions (store instructions) based at least in part on respective displacements between the read type and write type instructions (such as the load and store instructions exist in a loop, see Fig. 4-43) and on previous detection of respective aliasings between the read type instructions and the write type instructions, and that bypasses data (forward the store data directly to the load, see Fig. 4-43 (b)) from the write type instructions (store instructions) to the corresponding predicted to alias read type instructions (load instructions) using register information of the write type instructions.

Referring to claim 16, Shen et al. discloses, as claimed, a method comprising: in a register rename stage, tracking a write type instruction (store instruction) that has previously been indicated as aliased; and predicting (see Page 201, lines 9-14, regarding alias prediction) a read type instruction (load instructions) will alias with the write type instruction if displacement between the read type instruction and the write type instruction matches displacement between the read type instruction and a previously aliased write type instruction (such as the load and store instructions exist in a loop, see Fig. 4-43).

Referring to claim 28, Shen et al. disclose, as claimed, a method (see Fig. 4-46) comprising: observing repeated aliasing (such as the load and store instructions exist in a loop, see Fig. 4-43) between a first write type instruction (a first store instructions) and a read type instruction (load instructions) based at least in part on their static identifiers (such as the pointers for each program saved in a memory); determining a displacement (such as the load and store instructions exist in a loop, see Fig. 4-43) between the first write type instruction (store instructions) and the read type instruction based on dynamic identifiers (see Fig. 4-46, the addresses in store and load units) of the instructions; predicting aliasing (see Page 201, lines 9-14, regarding alias prediction) between the read type instruction (load instructions) as identified by the static identifier (such as the pointers for each program saved in a memory) thereof and a second write type instruction (a second store instruction) identified with a dynamic identifier (see Fig. 4-46, the addresses in store and load units) that corresponds to the read type instruction's dynamic identifier (see Fig. 4-46, the addresses in store and load units) and the displacement.

Referring to claim 38, Shen et al. disclose, as claimed, a method comprising: detecting aliasing (see Page 201, lines 9-14, regarding alias prediction) between a read type instruction and a first write type instruction; determining displacement (such as the load and store instructions exist in a loop, see Fig. 4-43) between the read type instruction and the first write type instruction, wherein the displacement is with respect to program execution; observing repeated aliasing between the read type instruction and the first write type instruction; selecting a second write type instruction based at

least in part on the displacement and the read type instruction; and bypassing data (forward the store data directly to the load, see Fig. 4-43 (b)) from the second write type instruction's data source to the read type instruction's data Destination (see claims 1 and 28 as set forth above).

Referring to claim 54, Shen et al. discloses, as claimed, an apparatus comprising: a data hazard detection module (see Fig. 4-46); and means for predicting (see Page 201, lines 9-14, regarding alias prediction) aliasing between a read type instruction (load instructions) and a first write type instruction (a first store instruction) based on displacement (such as the load and store instructions exist in a loop, see Fig. 4-43) between the instructions and previously observed aliasing between the read type instruction (load instructions) and a second write type instruction (a second store instruction), wherein the read type instruction (load instructions) and the second write type instruction also have the same displacement.

Referring to claim 62, Shen et al. disclose, as claimed, an apparatus comprising: an alias predictor (inside the Shen et al.'s system, see Page 201, lines 9-14, regarding alias prediction), including one or more encodings to host indications of particular instances of write type instructions (store instruction) and particular instances of read type instructions (load instructions), respective execution displacements (such as the load and store instructions exist in a loop, see Fig. 4-43) between respective ones of the particular instances of read and write type instructions, and register information of the particular write type instruction instances, to predict aliasings between read and write type instruction instances based, at least in part, on the encodings and indications of

detected aliasings between the instruction instances; a rename unit (inside the Shen et al.'s system, see Fig. 4-46) coupled with the alias predictor, the rename unit to supply register information for write type instruction instances to the alias predictor; and a data hazard detection unit (inside the Shen et al.'s system, see Fig. 4-46) coupled with the alias predictor, the data hazard detection unit to detect aliasing between particular instances of read and write type instructions (such as the load and store instructions exist in a loop, see Fig. 4-43) and to indication said detections to the alias predictor.

As to claim 2, Shen et al. also discloses: the processor of claim 1 wherein the data is bypassed (forward the store data directly to the load, see Fig. 4-43 (b)) if a threshold number of repeated aliases are detected between read type instructions and write type instructions with the respective displacements (as set forth, this is in the situation such as the load and store instructions exist in a loop, see Fig. 4-43).

As to claim 3, Shen et al. also discloses: the processor of claim 1 that includes encodings of read type instruction information, write type instruction information, and repeat aliasing (see Page 199, lines 10-11, regarding the additional priority encoding logic).

As to claim 10, Shen et al. also discloses: the processor of claim 1 wherein data bypasses comprise the processor substituting a register move instruction for the read type instruction (since Shen et al.'s system forwarding the store data directly to the load, see Fig. 4-43 (b)).

As to claim 11, Shen et al. also discloses: the processor of claim 10, wherein a loadCheck instruction is inserted, which when executed by the processor, causes the processor to verify the predicted aliasing (see page 199, lines 6-12, regarding the alias detection must be exact before forwarding of the data can be performed). Note LoadCheck is just a name without having any patentable weight.

As to claim 12, Shen et al. also discloses: the processor of claim 10 wherein the register move instruction includes an integer-to-integer move instruction, a floating point-to-floating point move instruction, an integer-to-floating point move instruction, and a floating point-to-integer move instruction (note the above are the situations of forwarding data between floating point-to-floating point, integer-to-floating point, and a floating point-to-integer).

As to claim 13, Shen et al. also discloses: the processor of claim 1 wherein data bypasses comprise the processor mapping the read type instruction's destination register to the write type instruction's source register (see Fig. 4-46 regarding Tag match).

As to claim 14, Shen et al. also discloses: the processor of claim 13 that replaces the read type instruction with a loadCheck instruction, which when executed by the processor, causes the processor to verify the predicted aliasing (see page 199, lines 6-12, regarding the alias detection must be exact before forwarding of the data can be performed). Note LoadCheck is just a name without having any patentable weight.

As to claim 15, Shen et al. also discloses: the processor of claim 14 wherein the processor's verification of the predicted aliasing comprises interrogation of a data

hazard detection module (inside the Shen et al.'s system, see page 199, lines 6-12, regarding the alias detection must be exact before forwarding of the data can be performed) to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match (see Fig. 4-46 regarding Tag match), and verification of absence of intervening matching write type instructions.

As to claim 17, Shen et al. also discloses: the method of claim 16 wherein the displacement is measured with dynamic instruction identifiers (see Fig. 4-46, the addresses in store and load units), wherein the dynamic instruction identifiers identify instances of instructions with respect to program execution.

As to claim 18, Shen et al. also discloses: the method of claim 16 wherein the write type instruction and the read type instruction are tracked with their static identifier (such as the pointers for each program saved in a memory), wherein the static identifier identifies an instruction in a program and remains static during program execution.

As to claim 19, Shen et al. also discloses: the method of claim 18 wherein the static identifier (such as the pointers for each program saved in a memory), includes an instruction address (since a pointer can be an address).

As to claim 20, Shen et al. also discloses: the method of claim 16 wherein a read type instruction (load instruction) includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction (note the above are just the different instruction types).

As to claim 21, Shen et al. also discloses: the method of claim 16 wherein the write type instruction (store instruction) includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction (note the above are just the different instruction types).

As to claim 22, Shen et al. also discloses: the method of claim 16 further comprising bypassing data (forward the store data directly to the load, see Fig. 4-43 (b)) of the write type instruction to the read type instruction with register information of the write type instruction.

As to claim 23, Shen et al. also discloses: the method of claim 22 wherein bypassing comprises mapping the read type instruction's destination register to the write type instruction's source register (see claim 13 as set forth above).

As to claim 24, Shen et al. also discloses: the method of claim 23 further comprising replacing the read type instruction with a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions (see claims 14 and 15 as set forth above).

As to claim 25, Shen et al. also discloses: the method of claim 22 wherein bypassing comprises converting the read type instruction to a register move instruction (see claim 10 as set forth above).

As to claim 26, Shen et al. also discloses: the method of claim 25 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection logic to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions (see claims 14 and 15 as set forth above).

As to claim 27, Shen et al. also discloses: the method of claim 16 embodied as a computer program product encoded in one or more machine-readable media (such as the main memory of the Shen et al.'s system).

As to claim 29, Shen et al. also discloses: the method of claim 28 further comprising bypassing data of the second write type instruction to the read type; instruction with register information of the second write type instruction (see claim 1 as set forth above).

As to claim 30, Shen et al. also discloses: the method of claim 29 wherein bypassing the data comprises mapping the data source of the read type instruction to the data source of the write type instruction (see claim 13 as set forth above).

As to claim 31, Shen et al. also discloses: the method of claim 30 further comprising substituting a loadCheck instruction for the read type instruction, wherein execution of the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the read type instruction and the second write type instruction match (see claims 14 and 15 as set forth above).

As to claim 32, Shen et al. also discloses: the method of claim 31 wherein execution of the loadCheck instruction further causes verifying the absence of intervening matching write type instructions (note this is certainly included in the steps for detecting or checking the instructions).

As to claim 33, Shen et al. also discloses: the method of claim 29 wherein bypassing the data comprises substituting a register move instruction for the read type instruction, wherein the move instruction moves data from the data source of the write type instruction to the data destination of the read type instruction (see claim 10 as set forth above).

As to claim 34, Shen et al. also discloses: the method of claim 28 wherein the dynamic identifiers (see Fig. 4-46, the addresses in store and load units) monotonically increase with execution of a program that includes the instructions (since the instruction addresses are allocated increasingly in a memory).

As to claim 35, Shen et al. also discloses: the method of claim 28 wherein the static identifiers include instruction addresses (see claim 19 as set forth above).

As to claim 36, Shen et al. also discloses: the method of claim 28 wherein the aliasing is predicted if the number of observed repeat aliasings exceeds a threshold (this is in the situation such as the load and store instructions exist in a loop, see Fig. 4-43).

As to claim 37, Shen et al. also discloses: the method of claim 28 embodied as a computer program product encoded in one or more machine-readable media (such as the main memory of the Shen et al.'s system).

As to claim 39, Shen et al. also discloses: the method of claim 38 further comprising verifying that the second write type instruction (the second store instruction) aliases with the first read type instruction (the first load instruction).

As to claim 40, Shen et al. also discloses: the method of claim 38 wherein data bypass comprises changing the read type instruction to a register move instruction (see claim 10 as set forth above).

As to claim 41, Shen et al. also discloses: the method of claim 40 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes verification that the read type instruction and the second write type instruction alias and verification of the absence of one or more intervening write type instructions (see claims 11 and 32 as set forth above).

As to claim 42, Shen et al. also discloses: the method of claim 38 wherein data bypasses comprise mapping the read type instruction's architectural destination register to the second write type instruction's rename source register (see claim 13 as set forth above).

As to claim 43, Shen et al. also discloses: the method of claim 42 further comprising changing the read type instruction to a loadCheck instruction, wherein the loadCheck instruction causes verification that the read type instruction and the second write type instruction alias and verification of the absence of one or more intervening write type instructions (see claims 11 and 32 as set forth above).

As to claim 44, Shen et al. also discloses: the method of claim 38 wherein the first and second write type instructions have a same static identifier (such as the

pointers for each program saved in a memory) and different dynamic identifiers (see Fig. 4-46, the addresses in store and load units since addresses in store and load units are changeable during the execution).

As to claim 45, Shen et al. also discloses: the method of claim 44 wherein the static identifiers (such as the pointers for each program saved in a memory) include instruction addresses (since a pointer can be an address).

As to claim 46, Shen et al. also discloses: the method of claim 38 wherein the displacement is based at least in part on the dynamic identifiers of the instructions, wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions (see claim 34 as set forth above).

As to claim 47, Shen et al. also discloses: the method of claim 38 embodied as a computer program product encoded in one or more machine-readable media (such as the main memory of the Shen et al.'s system).

As to claim 55, Shen et al. also discloses: the apparatus of claim 54 further comprising means for bypassing data (forward the store data directly to the load, see Fig. 4-43 (b)) from the write type instruction to the read type instruction with register information of the write type instruction.

As to claim 56, Shen et al. also discloses: the apparatus of claim 54 wherein the read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction (see claim 20 as set forth above).

As to claim 57, Shen et al. also discloses: the apparatus of claim 54 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction (see claim 21 as set forth above).

As to claim 63, Shen et al. also discloses: the apparatus of claim 62, wherein the indications of particular instances of instructions include instruction instance addresses (see Fig. 4-46, the addresses in store and load units; and such as the pointers for each program saved in a memory).

As to claim 64, Shen et al. also discloses: the apparatus of claim 63, wherein the instruction instances addresses include one or more of static identifiers (such as the pointers for each program saved in a memory) and dynamic identifiers (see Fig. 4-46, the addresses in store and load units).

Allowable Subject Matter

9. Claims 58-61 are allowed.

10. Claims 4-9, 65 and 66 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form.

11. The following is a statement of reasons for the indication of allowable subject matter: Shen et al., the closest reference, and the other prior art do not teach or fairly suggest: the instruction rename to track read type instructions indicated by the data hazard detection module as aliasing and repeat aliasing of the tracked read type instructions, and to indicate displacements between the tracked read type instructions and aliased write type instructions (in claim 58).

Response to Arguments

12. Applicant's arguments filed 8 September 2006 have been fully considered but they are not persuasive.

Applicant states:

> It should be obvious to the Examiner that the method claims fall within the process category of § 101. After determining that a claim falls under the process category of § 101, the Examiner should then determine whether the process is directed to nothing more than abstract ideas, such as mathematical algorithms, natural phenomena, or laws of nature. See Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility, Section IV. If a process claim is directed to an abstract idea, then the Examiner determines whether the claim falls within a judicially created exception requiring production of a useful, tangible, and concrete result. A reasonable and correct analysis cannot deem any of the method claims as directed to any one of mathematical algorithms, natural phenomena, or laws of nature. Observing or tracking repeated aliasing and then predicting aliasing based on displacement is not a mathematical algorithm, is not a natural phenomena, and is not a law of nature. Upon determining that the method claims are process category claims that are not directed solely to mathematical algorithms, natural phenomena, or laws of nature, then the analysis is complete and the Examiner should conclude that the claims are directed to statutory subject matter falling within the process category of § 101.

Examiner disagrees. Tracking and predicting do not have tangible results. A method of tracking instructions and making a prediction requires nothing tangible to occur. The claims do not recite limitations requiring action based on the tracking and

predicting. The claims merely require keeping track of an instruction and making a prediction. The prediction is made based on an algorithm based on many variables pertaining to the instructions. Since there is no useful result of the predicting (no action done), the process is directed to an algorithm for predicting. Examiner suggests adding a tangible result based on the prediction to the claims. Further, implementing the method as a program on a computer readable storage media does not remedy the problem. The media would not perform any useful and tangible actions.

13. Applicant states:

> To reject the independent claims 1, 16, 28, 38, 48, 54, and 62, the Examiner refers to Shen's disclosure of alias prediction at page 201 and Figure 4-43(b) of Shen. Since the Examiner repeatedly refers to this figure for many of the rejections, it is reproduced below...
>
> Examiner reasons, erroneously, that these two sections support the conclusion that Shen discloses an alias prediction technique that is based on displacement between instances of instructions. At page 201, Shen discloses possible use of a conventional alias predictor. See page 201, lines 8 - 13. Shen discloses that "[a] memory dependence predictor can be implemented to predict whether a load is likely to alias with its preceding stores." Shen never discloses that the predictor that "can be implemented" would base predictions on execution displacement. The Examiner refers back to Figure 4-43(b), which depicts forwarding store data for a "Store X" instruction to a "Load X" instruction, and cryptically states "such as the load and store instructions exist in a loop." This Figure depicts load forwarding, and does not depict alias prediction based on displacement. The corresponding text states that "if a trailing load aliases with a preceding store, i.e., there is a RAW dependence from the store to the load, load forwarding allows the load to receive its data directly from the store without having to access the data memory.... " Again, disclosure of predicting aliasing based on displacement is conspicuously absent. Shen discloses load forwarding and discloses that a predictor "can be implemented," but Shen never discloses or suggests predicting aliasing between a read type instruction and a write type instruction based on displacement as variously recited in the independent claims.

Examiner disagrees. The method of predicting aliasing as disclosed by Shen requires that both the store and load instruction be present in the pipeline. It is unreasonable to say that any processor would have an infinite instruction window.

Therefore, the prediction can only be made if the displacement of the load and store instructions is small enough so that the load can be executed first. For example, if the store and load instructions are a few thousand instructions apart, the processor would be unaware of the store instruction and the processor would not predict aliasing.

14. Applicant states:

> Shen fails to disclose or suggest bypassing data if a threshold number of repeat aliasings is detected
>
> Dependent claims 2 and 36 variously recite bypassing data if a threshold of repeated aliases is detected. To reject these claims, the Examiner simply refers to Figure 4-43(b). A picture of load forwarding does not disclose or suggest detecting a threshold number of repeat aliasings.

Examiner disagrees. Firstly, the claim language of claims 2 and 36 is extremely broad. The term "if" only imposes the limitation that if condition a, then outcome b. The limitation "a threshold number" includes all numbers, even number which would not be possible. Therefore, in these situations, the condition never occurs, therefore, the outcome need not happen. The claim does not recite any responsiveness to a condition and does not state that the condition must occur. Further, the "threshold number" could be 1. The prediction itself can be considered a repeat aliasing (since it predicts that the load with repeat the alias of the store). Therefore, if the instructions are predicted to store (threshold number = 1), the data is bypassed.

15. Applicant states:

> Shen fails to disclose or suggest replacing a read type instruction with a move instruction

> Dependent claims 10, 25, 33, 40, and 52 variously recite substituting a register move instruction for a predicted to alias instance of a read type instruction. Once again, the Examiner simply refers to Figure 4-43(b). Depicting load forwarding does not disclose or suggest substituting a move instruction for a predicted to alias read type instruction.

Examiner disagrees. The term "register move instruction" merely requires that the data be moved from the data source to the destination of the read instruction. As Applicant has pointed out, Shen discloses "if a trailing load aliases with a preceding store, i.e., there is a RAW dependence from the store to the load, load forwarding allows the load to receive its data directly from the store without having to access the data memory.... " Any operation which performs this functionality is considered to be the register move instruction.

16. Applicant states:

> Shen fails to disclose or suggest inserting a loadCheck instruction
>
> Dependent claims 11, 41, and 53 variously recite inserting a loadCheck instruction, which when executed, verifies a predicted alias. Shen discloses at page 199 verifying a detected alias before performing load forwarding. See page 199, lines 4 - 7. This disclosure by Shen does not disclose or suggest inserting an instruction into a program that will verify a predicted alias.

Examiner disagrees. The term "instruction" is not limiting to instruction words in an instruction stream. An instruction can broadly be interpreted as any piece of data controlling a processor. Therefore, any command internal to the processor which verifies a predicted alias is considered to be a loadCheck instruction.

17. Applicant states:

> Shen fails to disclose or suggest replacing an instance of a read type instruction with a loadCheek instruction
>
> Dependent claims 14, 24, 31, 43, and 51 variously recite replacing a predicted to alias instance of a read type instruction with a loadCheck instruction, which when executed, verifies a predicted alias. The Examiner again refers to Shen's disclosure of verifying a detected alias before performing load forwarding. See page 199, lines 4 - 7. This disclosure by Shen does not disclose or suggest replacing a predicted to alias instance of a read type instruction with a loadCheck instruction that will verify a predicted alias.

Examiner disagrees. As stated above, the term "instruction" is broadly interpreted as a command to control the processor. The control for verifying an alias is considered to be a loadCheck instruction.

Conclusion

18. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

19. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jesse R. Moll whose telephone number is (571)272-2703. The examiner can normally be reached on M-F 10:00 am - 6:30 pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Fritz M. Fleming can be reached on 571-272-4145. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

20. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Jesse R Moll
Examiner
Art Unit 2181

JM 11/26/06

FRITZ FLEMING
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100